REMARKS

Claims 1-15 are pending in the present application. By this Response, claims 6-15 are added. Claims 1, 4 and 5 are amended to recite "wherein each of the set of requester processes executes within the common interpreter process having the common process identifier and wherein each of the set of the requester processes shares the common process identifier." These features are supported on page 10, lines 5-10 of the current specification.

Reconsideration of the claims in view of the above addition to claims and the following remarks is respectfully requested.

I. 35 U.S.C. § 102(e), Alleged Anticipation, Claims 1, 4 and 5

The Office Action rejects claims 1, 4 and 5 under 35 U.S.C. § 102(e) as being anticipated by Jacobs et al. (U.S. Patent No. 6,334,114). This rejection is respectfully traversed.

As to claims 1, 4, and 5, the Office Action states:

> Referring to claim 1, Jacobs has taught a method for managing communications between requesters (items 202, 204, 206) and server processes (items 230, 234, 238) in a data processing network (Figure 2) including:
>
> creating a set of dispatcher processes (figure 2, items 214, 220, 226), each having a unique dispatcher process identifier (Col 25, lines 16-18);
>
> associating each of a set of requester processes, which communicate with a server process via a common interpreter process having a common process identifier (Col 8, lines 57-59, Col 20, lines 24-67), with a unique dispatcher process of said set of dispatcher processes (Col 25, lines 16-18);
>
> for requests sent from any of said set of requester process via said common interpreter process to server process which identifies each of said set of requester processes using the unique dispatcher process identifier, routing said requests via the associated dispatcher process (Col 25, lines 16-25);
>
> at the respective dispatcher process, attaching the unique dispatcher process identifier to the request and then forwarding the request to the server process (Col 25, lines 16-25);
>
> responsive to receipt by the dispatcher process of a reply to said request, forwarding the reply to the associated requester process via the common interpreter process (Col 8, lines 60-63).

Final Office Action dated March 1, 2004, pages 2-3.

Amended independent claim 1, which is representative of independent claims 4 and 5 with regard to similarly recited features, now recites:

> 1. A method for managing communications between requester processes and server processes in a data processing network, including:
>
> creating a set of dispatcher processes, each have a unique dispatcher process identifier;
>
> associating each of a set of requester processes, which communicate with a server process via a common interpreter process having a common process identifier, with a unique dispatcher process of said set of dispatcher processes, <u>wherein each of the set of requester processes executes within the common interpreter process having the common process identifier and wherein each of the set of requester processes shares the common process identifier;</u>
>
> for requests sent from any of said set of requester processes via said common interpreter process to a server process which identifies each of said set of requester processes using the unique dispatcher process identifier, routing said requests via the associated dispatcher process;
>
> at the respective dispatcher process, attaching the unique dispatcher process identifier to the request and then forwarding the request to the server process; and
>
> responsive to receipt by the dispatcher process of a reply to said request, forwarding the reply to the associated requested process via the common interpreter process.
>
> (emphasis added)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. Kalman v. Kimberly-Clark Corp., 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicant respectfully submits that Jacobs does not teach every element of the claimed invention arranged as they are in claim 1. Specifically, Jacobs does not teach that <u>each of the set of requester processes executes within the common interpreter process having the common</u>

process identifier and wherein each of the set of requester processes shares the common process identifier, as recited in claim 1.

Jacobs teaches a method and system for processing multiple-request transactions in stateless environment. A cartridge execution engine intercepts browser messages directed to a cartridge and determines whether the browser messages are associated with transactions. The cartridge execution engine then sends transaction control messages to a transaction manager. In addition, the cartridge execution engine sends operation messages to the cartridge. Then cartridge then performs the operation specified in the operation messages. In response to the transaction control messages from the cartridge execution engine, the transaction manager causes the multiple-request transactions to be either committed or rolled back as an atomic unit of work (Abstract).

However, Jacobs does not teach that each of a set of requester processes executes within a common interpreter process having a common process identifier and wherein each of the set of requester processes shares the common process identifier, as recited in independent claim 1. At column 6, lines 47-56, Jacobs teaches the functional overview of the application server, which reads as follows:

> A typical operation within system **200** generally includes the following stages:
>
> A browser transmits a request over the Internet **208**.
>
> A listener receives the request and passes it through a transport adapter to a dispatcher.

In the above section, a browser sends a request (a requester process) over the Listener, which in turn passes the request (the requester process) to a dispatcher. At column 8, lines 15-35, Jacobs teaches that dispatchers **214**, **220** and **226** are associated with listeners **210**, **216** and **222**. Dispatchers **214**, **220** and **226** selectively route browser requests received by listeners **210**, **216** and **222** to cartridges. The request received by listener **210** over the Internet **208** is delivered in a form of a Uniform Resource Locator (URL) and that the browser request serves as an identifier for a Web object, for example, an HTML page or an operation to be performed. The listener **210** hands off the browser request to dispatcher **214** without any attempt at interpreting the browser request.

Thus, according to Jacob's teaching, each browser request (a requester process) is handled by a different listener, which forwards the request to a corresponding dispatcher. While dispatchers and listeners are executing within the same environment in Web application server **200**, there is no teaching or suggestion in Jacobs that the environment has a common process identifier. There is no mention of a common process identifier in the reference. In addition, there is no mention of a common interpreter process that has a common process identifier that is shared among each of the browser requests.

To the contrary, Jacobs teaches, at column 20, lines 24-67, a global unique transaction ID that is used to identify the multiple-request transaction to which the browser request belongs, which reads as follows:

> For example, when a browser **202** sends a first browser request associated with a transaction and a begin transaction URL, the transaction manager **606** creates a unique browser identifier and sends it to the dispatcher **214**. The dispatcher **214** then causes the global unique transaction ID to be stored as cookie information on browser **202**. When browser **203** sends a second browser request that is associated with the same transaction, the dispatcher **214** obtains the globally unique transaction ID contained in the cookie information of browser **202**.
>
> Using the globally unique transaction ID, the database servers that ultimately process the browser request can determine both the first browser request and the second browser request are associated with the same multiple-request transaction. Because a particular browser may be executing more than one transaction at a time, in certain embodiments, the cartridge name for the particular transaction is contained within each globally unique transaction ID and is used to help identify the particular transaction to which the globally unique transaction ID corresponds.

Thus, the global unique transaction ID only identifies a particular transaction with which multiple browser requests from a given browser are associated. The global unique transaction ID has nothing to do with the environment within which each of the browser requests executes, let alone a common interpreter process that has a common process identifier. The global unique transaction ID does not identify a common interpreter process within which each of the set of the requester processes (browser requests) executes. Since Jacobs does not teach a common interpreter process that has a common process identifier, Jacobs also does not teach that each of a set of requester processes shares the common process identifier.

In view of the above, Applicant respectfully submits that Jacobs does not teach or suggest each and every feature recited in independent claim 1. Claims 4 and 5 recite similar features also not taught by Jacobs. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 1, 4 and 5 under 35 U.S.C. § 102(e).

II. 35 U.S.C. § 103(a), Alleged Obviousness, Claim 2-3

The Office Action rejects claims 2 and 3 under 35 U.S.C. § 103(a) as being unpatentable over Jacobs et al. (U.S. Patent No. 6,334,114) in view of Bayeh (U.S. Patent No. 6,223,202 B1). This rejection is respectfully traversed.

As to claims 2-3, the Office Action states:

> Referring to claim 2 -3, recites the limitation of a using Java Virtual Machine including respective Servlet running on a Web server, and Jacobs has not taught such limitation using Java Virtual Machine.
>
> However, all the claimed elements, such as JVM, Web server, Web application server, Servlet threads, Web browser are all well known network communication elements in the network communication arts. Bayeh shows a "A Web server that implements a Java virtual machine can be functionally extended using Java "servlets." (Col 2 lines 27-35), and also teaches dispatchers which communicates with web servers under the virtual machine environment (Col 7 lines 41-59).
>
> It would have been obvious to a person with ordinary skill in the art at the time the invention was made to modify the teaching of Jacobs such that to use a Java Virtual Machine including respective Servlet running on a Web server in his invention because Bayeh has taught a dispatcher which communicates with web servers under the virtual machine environment (Col 7 lines 41-59).
>
> A person with ordinary skill in the art would have been motivated to make the modification to Jacobs because having JVM running in Jacob's system would allow the Java programming language to be used in Jacob's invention, and Java is gaining a wide acceptance for writing Web applications, as it is a robust portable object-oriented language defined specifically for the web environment (Col 2 lines 1-26).

Final Office Action dated March 1, 2004, pages 4-5.

Dependent claim 2 reads as follows:

2. A method according to claim 1, wherein the common interpreter process via

which each of said set of requester processes associated with the unique dispatcher process communicates is a Java Virtual Machine.

As discussed in arguments previously presented for claim 1, Jacobs does not teach each of a set of requester processes executes within a common interpreter process having the common process identifier and wherein each of the set of requester processes shares the common process identifier. Bayeh also does not teach these features.

Bayeh teaches a system for enabling multiple virtual machines to execute on a single server, using virtual machine pooling. The integrity of an application's data is protected from inadvertent overwriting by another application, because each application can be running in a separate virtual machine. Garbage collection, crashes, and hangs do not temporarily or completely halt a server: when one virtual machine halts, other can continue executing. Multiple environments can execute on a single server, including different versions of virtual machines, increasing the mix of servlets that can be supported. Further, debugging can occur concurrently with normal application execution, by isolating the debugging function to a specific virtual machine (Abstract).

Bayeh does not teach the features as recited in amended claim 1. Bayeh teaches in **Figure 4** that multiple virtual machines are pooled in a single Web server. Each of the virtual machines has a listener servlet running on it, which takes a request from a queue and assigns the request to an available servlet running on the virtual machine. However, Bayeh does not teach that each virtual machine has a common process identifier shared by each of the servlets running on it.

To the contrary, at column 8, lines 35- 51, Bayeh teaches that when the Web server gets the request, the request is forwarded to the dispatcher. The request contains a URL that identifies the target machine for processing this request. As part of the process of configuring the virtual machine pool, one or more URL masks will have associated with each virtual machine. The dispatcher is responsible for inspecting the URL of each incoming request, and comparing the URL to the URL masks for each virtual machine. In this way, the dispatcher determines which virtual machine to forward the request to. Thus, rather than using a common process identifier that identifies a common interpreter process within which each of the set of requester processes executes, Bayeh uses a URL to identify a particular virtual machine in order to forward a request. Therefore, Bayeh does not teach the features of claim 1, from which claims 2 and 3 depend.

In addition, Bayeh does not teach that the common interpreter process via which each of said set of requester processes associated with the unique dispatcher process communicates is a Java Virtual Machine, as recited in claim 2. The Office Action alleges that Bayeh teaches these features at column 2, lines 27-35, and at column 7, lines 41-49, both sections read as follows:

> A Web server that implements a Java Virtual Machine can be functionally extended using Java "servlets". A servlet is a relatively small executable code object that can be dynamically plugged in, or added, to the code running on the server. Servlets typically perform some specialized function, which can be invoked by the server (or by another servlet) to extend its own functionality. The servlet processes the request, and returns the response to the server (or servlet) that invoked it.

(Column 2, lines 27-35, Bayeh)

> As shown in **FIG. 4**, the server **60** has a plug-in **151** running on it. (A plug-in is executable code that extends the functionality of the Web server, and is merely one form in which this component of the preferred embodiment may be packged.) This plug-in **151** includes a dispatcher component **149**, which receives client requests **110** from the Web server **60**, and then route those requests to one of the virtual machines **152**, **154** and **156**. The requests are received into an application queue, where they remain until assigned to a servlet for execution.

(Column 7, lines 41-49, Bayeh)

However, in the above sections, Bayeh merely teaches a Java Virtual Machine, the functionality of which may be extended using servlets and plug-in applications. Bayeh does not teach the JVM as a common interpreter process via which each requester process is associated with a unique dispatcher process, as recited in claim 2. The JVM recited in claim 2 allows each of the requester processes to be executed within the same JVM in order to communicate with the server process.

To the contrary, Bayeh solves the problem differently by allowing multiple JVMs to be executed within the same Web server. Each of the servlets may be executed on different JVMs. In this way, each of the requester processes communicates with a server process via different virtual machines, as opposed to a common interpreter process. Bayeh also fails to mention a common process identifier shared by each of the servlets.

In addition, Bayeh does not teach each of the set of requester processes is associated with the unique dispatcher process, as recited in amended claim 2. To the contrary, Baych teaches only one dispatcher component in **Figure 4**, which is used to route requests to one of the application queue in virtual machines **152, 154, 156**. Thus, each of the requester processes in Bayeh is associated with the same dispatcher plug-in **149**, as opposed to a unique dispatcher process. Therefore, Baych does not teach features specific to claim 2.

Furthermore, a person of ordinary skill in the art would not have been led to modify Jacob's system to include Bayeh's teaching to reach the presently claimed invention, because Jacobs is only concerned with a system that allows multiple-requests for a given browser to be associated with a global unique transaction ID, Jacobs is not concerned with allowing browser requests to execute within a common interpreter process that has a common process identifier shared by each of the browser requests. Baych is only concerned with running multiple servlets on multiple Java virtual machines. Bayeh is not concerned with running multiple servlets in a common interpreter process having a common process identifier, which is shared by each of the servlets.

Moreover, even though a person of ordinary skill in art is to combine both references, the result would not be the same as the presently claimed invention. The result would be a system that has multiple virtual machines executing a number of requester processes. Each of the requester processes would not be executing within a common interpreter process and sharing a common process identifier.

Therefore, in view of the above, Applicant respectfully submits that neither Jacobs nor Baych, either alone or in combination, teaches or suggests each and every feature recited in dependent claim 2. By virtual of its dependency on claim 2, neither Jacobs nor Bayer, either alone or in combination, teaches or suggests each and every feature in dependent claim 3. Thus, Applicant respectfully requests withdrawal of the rejection of claims 2 and 3 under 35 U.S.C. § 103(a).

III. Newly Added Claims, Claims 6-15

Claims 6 and 11 are added to recite "wherein associating each of a set of requester processes with a unique dispatcher process of said set of dispatcher processes includes invoking a service module to request a unique dispatcher process for each of the set of requester processes." These features are supported at least on page 12, lines 15-17 of the current specification.

Claims 7 and 12 are added to recite "wherein the service module creates a unique dispatcher process for each of the set of requester processes and assigns each unique dispatcher process to each of the set of requester processes." These features are supported at least on page 12, lines18-20 of the current specification.

Claims 8 and 13 are added to recite "wherein the service module maintains a table of relationships between each of the requester processes and assigned unique dispatcher process." These features are supported at least on page 12, line 21 of the current specification.

Claims 9 and 14 are added to recite "wherein the service module is invoked by calling a register method implemented by each of the set of requester processes." These features are supported at least on page 12, line 15 of the current specification.

Claims 10 and 15 are added to recite "wherein routing said requests via the associated dispatcher process includes invoking a send method implemented by a native dynamically linked library." These features are supported at least on page 13, lines 1-2 of the current specification.

IV. Conclusion

It is respectfully urged that the subject application is patentable over Jacobs et al. (U.S. Patent No. 6,334,114) in view of Bayeh (U.S. Patent No. 6,223,202 B1) and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: June 1, 2004

Stephen J. Walder

Stephen J. Walder, Jr.
Reg. No. 41,534
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Attorney for Applicant

SJW/im